## CLAIMS

What is claimed is:

546 A1

1. A method of generating code representing a mapping between a source schema and a target schema, the mapping comprising data transformations from the source schema to the target schema, the source schema comprising a source tree having a source node and the target schema comprising a target tree having a target node, the method comprising:

determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema;

matching hierarchy by generating a hierarchy match list for the target node; and

generating code according to the hierarchy match list.

2. The method of claim 1, further comprising:

initializing node dependencies memory comprising:

allocating memory for a compiler node;

associating the compiler node with the target node;

allocating memory for compiler variable classes; and

associating compiler variable classes with functoids.

3. The method of claim 2, wherein determining source node dependencies comprises:

creating a target dependencies list for the target node;

creating a source dependencies list for the target node;

tracing back to the source tree through all possible paths recursively, adding source tree nodes to corresponding source dependencies lists;

setting a flag for the compiler nodes associated with the target node and an associated parent compiler node if there is a path from the target tree node to the source tree;

Sub A'7

adding the target tree node to the target dependency list of an associated parent record node if the target node is a field node;

consolidating the target dependency list for target record nodes;

consolidating the source dependency list for target record nodes;

selecting a source loop path node for the target node according to the consolidated source dependency list; and

storing the source loop path node in the compiler node associated with the target node.

4. The method of claim 3, wherein consolidating the target dependency list for target record nodes comprises adding child record nodes to the target dependency list if the target record node has a child field node with a flag set and the target record node has a child record node with a flag set, and wherein consolidating the source dependency list for target record nodes comprises adding source dependencies of each node in the target dependency list to the target dependency list.

5. The method of claim 4, wherein selecting a source loop path node for the target node according to the consolidated source dependency list comprises:

iterating through source nodes in the source dependency list;

finding the highest level loop point for which more than one occurrence of the source node is possible;

storing the current node in the source dependency list as the source loop path node if no loop point is found;

storing the current node in the source dependency list as the source loop path node if the current loop point is at a higher level than the previous loop point and if the paths to the previous and current loop points converge;

storing the previous node in the source dependency list as the source loop path node if the current loop point is at a lower level than the previous loop point and if the paths to the previous and current loop points converge;

Sub A17

storing the current node in the source dependency list as the source loop path node if the current loop point is at the same level as the previous loop point, and if the paths to the previous and current loop points converge;

generating a compiler error if the paths to the previous and current loop points do not converge.

6. The method of claim 5, wherein finding the highest level loop point for which more than one occurrence of the source node is possible comprises finding the highest level loop point for which maxoccurs=*.

7. The method of claim 5, wherein storing the source loop path node in the compiler node associated with the target node comprises:

storing the source loop path node as the source loop path node in the compiler node if the source loop path node is a record node; and

storing the parent record node of the source loop path node as the source loop path node in the compiler node if the source loop path node is a field node.

8. The method of claim 7, wherein matching hierarchy by generating a hierarchy match list for the target node comprises:

creating a hierarchy match list for the target node if the target node has a source loop path node;

determining a consolidated compiler link option using links on target node dependencies; and

matching hierarchy according to the consolidated compiler link option.

9. The method of claim 8, wherein matching hierarchy according to the consolidated compiler link option comprises:

climbing up the source tree from the source loop path node, adding each source node to the hierarchy match list for the target node if the compiler link option is flattening;

Sub A7

traversing up the source tree until the source tree node level matches the target node level, adding source tree nodes to the hierarchy match list if the target node level is greater than the source loop path node level and if the compiler link option is top-down;

traversing up the target tree until the source tree node level matches the target node level, adding the source loop path node to the hierarchy match list of each target node traversed if the source loop path node level is greater than the target node level and if the compiler link option is top-down;

traversing up the source and target trees, adding source tree nodes to the hierarchy match lists of target nodes level by level if the compiler link option is top-down;

traversing up the source and target trees, adding source nodes to the hierarchy match lists of target nodes until level 1 of either the source or the target tree is reached if the compiler link option is bottom-up;

traversing up the source tree, adding the source tree node to the hierarchy match lists of the target tree nodes if the compiler link option is bottom-up and if level 1 of the target tree is reached; and

traversing up the source tree, adding the source tree nodes to the hierarchy match list of the target node.

10. The method of claim 9, wherein generating code according to the hierarchy match list comprises:

generating a code header for a root node;

generating a code trailer for the root node;

processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function; and

processing field nodes in the executeleaf function.

11. The method of claim 10, wherein processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function comprises:

Sub A17

generating <xsl : for-each> code in the preexecuteparent function using the hierarchy match list;

generating <xsl : if> code in the preexecuteparent function if there is an incoming link from a conditional functoid;

generating a start tag code for the target record node in the preexecuteparent function;

generating </xsl : if> code in the postexecuteparent function for record nodes if there is an incoming link from a conditional functoid;

generating <xsl : value-of> code in the postexecuteparent function for record nodes if the incoming link is not from a conditional link by traversing a grid to source nodes, and generating code for constant node values;

generating </xsl : for-each> code in the postexecuteparent function for record nodes using the hierarchy match list;

generating <xsl : for-each> code in the executeleaf function for leaf record nodes using the hierarchy match list;

generating <xsl : if> code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating a start tag code in the executeleaf function for leaf records;

generating </xsl : if> code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating <xsl : value-of> code in the executeleaf function for leaf record nodes if the incoming link is not from a conditional link by traversing the grid to source nodes, and generating code for constant node values; and

generating <xsl : value-of> code in the executeleaf function for field nodes by traversing grid to source nodes, and generating code for constant node values.

12. The method of claim 11, wherein generating <xsl : value-of> code comprises:

using the value of a source tree node to generate code if the link is simple; and

Sub A17

tracing back to the source tree through all possible paths recursively and generating <xsl : variable name=...select=.../> code for each functoid if there is a functoid link.

13. The method of claim 12, wherein determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema comprises depth-first tree traversal of the target tree, matching hierarchy by generating a hierarchy match list for the target node comprises depth-first tree traversal of the target tree, and wherein generating code according to the hierarchy match list comprises field-attribute tree traversal of the target tree.

14. The method of claim 10, wherein processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function comprises:

generating looping start code in the preexecuteparent function using the hierarchy match list;

generating conditional start code in the preexecuteparent function if there is an incoming link from a conditional functoid;

generating a start tag code for the target record node in the preexecuteparent function;

generating conditional end code in the postexecuteparent function for record nodes if there is an incoming link from a conditional functoid;

generating value code in the postexecuteparent function for record nodes if the incoming link is not from a conditional link by traversing a grid to source nodes, and generating code for constant node values;

generating looping end code in the postexecuteparent function for record nodes using the hierarchy match list;

generating looping start code in the executeleaf function for leaf record nodes using the hierarchy match list;

generating conditional start code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

Sub A7

generating a start tag code in the executeleaf function for leaf records;

generating conditional end code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating value code in the executeleaf function for leaf record nodes if the incoming link is not from a conditional link by traversing the grid to source nodes, and generating code for constant node values; and

generating value code in the executeleaf function for field nodes by traversing grid to source nodes, and generating code for constant node values.

15. The method of claim 2, wherein matching hierarchy by generating a hierarchy match list for the target node comprises:

creating a hierarchy match list for the target node if the target node has a source loop path node;

determining a consolidated compiler link option using links on target node dependencies; and

matching hierarchy according to the consolidated compiler link option.

16. The method of claim 2, wherein generating code according to the hierarchy match list comprises:

generating a code header for a root node;

generating a code trailer for the root node;

processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function; and

processing field nodes in the executeleaf function.

17. The method of claim 16, wherein processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function comprises:

generating <xsl : for-each> code in the preexecuteparent function using the hierarchy match list;

Sub A7

generating <xsl : if> code in the preexecuteparent function if there is an incoming link from a conditional functoid;

generating a start tag code for the target record node in the preexecuteparent function;

generating </xsl : if> code in the postexecuteparent function for record nodes if there is an incoming link from a conditional functoid;

generating <xsl value-of> code in the postexecuteparent function for record nodes if the incoming link is not from a conditional link by traversing a grid to source nodes, and generating code for constant node values;

generating </xsl : for-each> code in the postexecuteparent function for record nodes using the hierarchy match list;

generating <xsl : for-each> code in the executeleaf function for leaf record nodes using the hierarchy match list;

generating <xsl : if> code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating a start tag code in the executeleaf function for leaf records;

generating </xsl : if> code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating <xsl : value-of> code in the executeleaf function for leaf record nodes if the incoming link is not from a conditional link by traversing the grid to source nodes, and generating code for constant node values; and

generating <xsl : value-of> code in the executeleaf function for field nodes by traversing grid to source nodes, and generating code for constant node values.

18. The method of claim 1, wherein generating code according to the hierarchy match list comprises creating an XSL style sheet representation of the mapping.

19. The method of claim 16, wherein processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function comprises:

Sub A7

generating looping start code in the preexecuteparent function using the hierarchy match list;

generating conditional start code in the preexecuteparent function if there is an incoming link from a conditional functoid;

generating a start tag code for the target record node in the preexecuteparent function;

generating conditional end code in the postexecuteparent function for record nodes if there is an incoming link from a conditional functoid;

generating value code in the postexecuteparent function for record nodes if the incoming link is not from a conditional link by traversing a grid to source nodes, and generating code for constant node values;

generating looping end code in the postexecuteparent function for record nodes using the hierarchy match list;

generating looping start code in the executeleaf function for leaf record nodes using the hierarchy match list;

generating conditional start code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating a start tag code in the executeleaf function for leaf records;

generating conditional end code in the executeleaf function for leaf record nodes if there is an incoming link from a conditional functoid;

generating value code in the executeleaf function for leaf record nodes if the incoming link is not from a conditional link by traversing the grid to source nodes, and generating code for constant node values; and

generating value code in the executeleaf function for field nodes by traversing grid to source nodes, and generating code for constant node values.

20. A method for compiling a mapping between a source schema having source nodes associated therewith, and a target schema having target nodes associated therewith, comprising:

determining source node dependencies for at least one target node by tracing from the at least one target node through the mapping to the source schema;

matching hierarchy by generating a hierarchy match list for the at least one target node; and

generating code according to the hierarchy match list.

21. The method of claim 20, further comprising:

initializing node dependencies memory prior to determining source dependencies; and

freeing node dependencies memory after generating code.

22. The method of claim 20, wherein determining source node dependencies comprises picking a source loop path for each target node, and detecting multiple source loop paths.

23. The method of claim 20, wherein determining source node dependencies comprises generating a source dependency list.

24. The method of claim 22, further comprising generating a compiler error if multiple source loop paths are detected.

25. The method of claim 20, wherein matching hierarchy comprises generating a hierarchy match list.

26. The method of claim 20, wherein matching hierarchy comprises one of flattening, top-down, and bottom-up matching.

27. The method of claim 20, wherein the source schema and the target schema are XML schemas.

28. The method of claim 20, wherein generating code comprises creating an XSL style sheet representation of the mapping.

Sub A7

29. A system for generating code representing a mapping between a source schema and a target schema, the mapping comprising data transformations from the source schema to the target schema, the source schema comprising a source tree having source nodes and the target schema comprising a target tree having target nodes, the system comprising:

means for determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema;

means for matching hierarchy by generating a hierarchy match list for the target node; and

means for generating code according to the hierarchy match list.

30. A computer-readable medium having computer-executable instructions for:

generating code representing a mapping between a source schema and a target schema, the mapping comprising data transformations from the source schema to the target schema, the source schema comprising a source tree having a source node and the target schema comprising a target tree having a target node;

determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema;

matching hierarchy by generating a hierarchy match list for the target node; and

generating code according to the hierarchy match list.